NoSQL Write-up – Dota2 Game Replay Analysis

Team name: PSG.LGD

Yichun Yan
Ziwei Jiang
Yifan Li
Weiqi Wang

October 26, 2019

Contents

1 Datasets

1.1 Background of Datasets

Our team aims at exploring data about a popular and long-lived computer game, Dota2. Dota2 is a multiplayer online battle arena (MOBA) video game developed and published by Valve Corporation. Each game will involve ten players that are divided to two teams play against each other on a same map. Every player chooses one "hero" that has unique abilities and different styles of playing. The aim for each team is to destroy the building called "Ancient" of the other team. Players collect gold to buy powerful items and gain XP by killing players on the other team, destroy building of the other team, or kill creeps.

Figure 1: A screenshot of the game

It is a real-time game, hundreds of operations and some decisions will be made each second, and all of them will have some influence on the final results of the game. Therefore, there are some professional teams start hiring data engineers to help their players perform better in a game. For example, by using big data pipeline to answer the business question "Does there exist a regular farming path for some professional players?", we can understand both our teammates and the players from other team better, and therefore optimize our strategies.

1.2 Source of Datasets

Dataset Name	Dataset Source
Dota2 match result dataset	Valve's official API
Dota2 replay dataset	Valve's Dota2 replay servers

1.3 Descriptions of Datasets

Dataset Name	Format	Usage
Dota2 match result dataset	JSON	• Get match result (containing important information of players and the game such as player name, level, first blood time etc.) • Provide keys for replay dataset
Dota2 replay dataset	`.dem.bz2` binary file	• Get player name, id, team formation and hero choice • Get detail log of the game • Get spawn/death of heros and NPCs

We will describe how we use Dota2 replay parser to transform different datasets format into usable format in the ETL section.

1.4 Data Dictionaries

2 Data ETL

2.1 Extraction

2.2 Transformation

2.3 Loading

2.3.1 Docker

As mentioned in the proposal, we tried to containerized the project by docker. It is convenient to split up the works for every team mates if we containerized our project. By learning from documents, I tried to configure our project on docker but it ends up with failures. Considering the limited time, we decided to discard using docker for installing our NoSQL technology. However, we learned a lot from this experience.

1. Getting Started

   At this phase, I installed docker, read docker documents, explored docker hub and played around with docker. The objective of this phase is to get familiar with docker and in the end find the way to build the project in Docker way. I learned that we need to build an image, which is an executable package to run the project in the environment of docker.

   - This ”package” contains:
     - A runtime as a parent image, the overall environment for the container launched by this image; (i.e. `FROM ubuntu:18.04` from Dockerfile . line2)
     - Our project code; (i.e. `RUN java -jar target/FetchStore.one-jar.jar 4182489531 10 100` from Dockerfile . line52)

  - How we define environment variables;
    (i.e. `ENV JAVA_HOME /usr/lib/jvm/java-8-openjdk-amd64/` from Dockerfile . line23)
  - Installed packages such as java8, maven, mongodb, as the prerequisite of Data ETL;
  - Working directory set up;
    (i.e. `WORKDIR /app` from Dockerfile . line44)

  By launching this image, we can get a running container. In other words, a container is an instance of the image. And then we can share our images in a repository on docker hub. When this is done, my teammates can pull and run the image remotely.

2. Dockerfile

   According to the docker documents, Dockerfile "defines what is going on in the environment inside your container". At the very beginning, I thought dockerize a project just need to pull all the official images in docker, and then we can start manipulate it. But then, after reading documents, I found that we need to construct a Dockerfile to define how a project work, so that it can behaves exactly the same wherever it runs. We can pull an image as a parent image, and then install needed packages in it.

   For example, in the project, I use ubuntu as a parent image, then docker virtualize an ubuntu operation system environment on your host machine, and then I use Dockerfile as if it is the terminal of ubuntu, writing codes of installing java8, maven, mongodb in the Dockerfile. As mentioned above, then I can use `RUN` command to run project code.

   After I constructed the Dockerfile and try to build the project using `$docker build .` in command line, I failed every time when it trying to connect to the database. After reading documents and google for reasons, I realized that it is cramped to put all the service on a Dockerfile. I should construct a docker-compose.yml to define, run and scale service with Docker.

3. docker-compose.yml

   According to our data pipeline, I intended to build 4 services in the docker-compose:

   - Data acquisition and filtering (as "fetchstore" service in the docker-compose, line 17-24)
   - Data Extraction (as "parsereplay" service in the docker-compose, line 27)
   - Data Aggregation and Representation (as "mongodb" service in the docker-compose, line 4-16)
   - Data Analysis (as "analytics" service in the docker-compose, line 29-32)

   In the docker-compose, we can also use the Dockerfile using the command `build` in a service. Therefore, my thought is to construct Dockerfile for "fetchstore", "paresereplay", and "analytics" service using the code and configuration file we have. And use the official image of mongodb, using `volume` in the service to reflect the direction of configuration file and mongodb initial javascipt (mogo-init.js, a file to define initial user, database, collections). Also, we used `depends_on` to define simple interaction between the services.

As the result, we still have a hard time connecting the database. When the problem is fixed, we only finished the configuration of the "fetchstore" service and the "mongodb". Then we realized we got limited time left, therefore, we discard docker for now, and move on to deploy our project on amazon web service (AWS). And we hope to complete this part in the future.

The reason of failure is that I got misunderstanding on Docker's architecture. Also, it is hard for me to consider our system's architecture in Docker's way. And it also need many times of failure to realize the right way to deploy a project on Docker. This experience help us learn how to make the best of documents and google, and also push me to learn a lot about Linux and to read a lot of configuration file.

3 Data Analysis

4 Challenges

Glossary

Ancient (also commonly refered to as Thrones, or Tree for Radiant's ancient and Throne for Dire's ancient, as legacy names from DotA) are massive structures found inside each faction's base and are the main objective. In order to win, the enemy team's Ancient must be destroyed, while the own one must be kept alive. Ancients are guarded by their two tier 4 towers. The Ancients are invulnerable until both of their tier 4 towers are destroyed.. 2

creeps Creeps are basic units in Dota 2. Every unit which is not a hero, building, ward or courier is considered a creep. Creeps can belong to either faction, be neutral, or be player-controlled units. Unlike heroes, creeps do not gain experience and cannot level up. All of their stats are set values (though can still be altered by modifiers). Most creeps grant a set gold and experience bounty to heroes when killed.. 2

gold Gold is the currency used to buy items or instantly revive your hero. Gold can be earned from killing heroes, creeps, or buildings.. 2

MOBA also known as action real-time strategy (ARTS), is a subgenre of strategy video games that originated as a subgenre of real-time strategy in which each player controls a single character, usually on a map in an isometric perspective, as part of a team competing against another team of players.. 2

XP A shorthand for experience. Experience is an element heroes can gather by killing enemy units, or being present as enemy units get killed. On its own, experience does nothing, but when accumulated, it increases the hero's level, so that they grow more powerful. Only heroes can gather experience and therefore reach higher levels. With each level gained, a hero's base attributes increase by static values (unique for each hero), which makes them stronger in several.. 2